Content Processing Platform

Installation Guide

Compliance Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received including interference that may cause undesired operation.

WARNING: This equipment has been tested and found to comply with the limits for a Class "A" digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at their own expense.

Changes or modifications not expressly approved by Tarari, Inc. could void the user's authority to operate the equipment.

Legal Information

Tarari Contact Information

Additional information: info@tarari.com
Internet: http://www.tarari.com/
Telephone: (858) 385-5131
Fax: (858) 385-5129
Documentation questions or comments: documentation@tarari.com

Tarari, Inc.
10908 Technology Place
San Diego, CA 92127-1874
USA

A02202-001

Contents

1 CPP Installation and Overview

Purpose

This document describes how to install a Tarari Content Processing Platform (CPP) board.

Installing the CPP PCI Board

1. If your system is on, shut it down, and remove the AC power cord.
2. Open your system.
3. Ensure the orientation of the connector is correct, and install the CPP board into an available 64-bit, 3.3v PCI slot, preferably one that has enough room for you to view the LEDs on the front and rear of the board. For more information, see chapter 4.
4. Reconnect the AC power cord.
5. Restart your system.

Where to Go from Here

Windows*

For Windows installations, see chapter 2:

Item	Where to find it
Base driver	"Installing the CPP Base Driver" on page 6
Bitstream to Agent (Signing) Utility	"Using the Bitstream to Agent (Signing) Utility" on page 7
Agent Configuration Tool	"Using the Configuration Tool" on page 14
Diagnostic Agent Driver	"Running Diagnostics" on page 19

Table 1: Windows Installation Highlights

Linux

For Linux installations, see chapter 3:

Feature	Where to find it
Base Driver and Diagnostic Agent	"Installing the Base and Diagnostic Agent Drivers" on page 22
Bitstream to Agent (Signing) Utility	"Using the Bitstream to Agent (Signing) Utility" on page 23
Linux CPP Board Configuration	"Using the cpp_agent_manager Tool" on page 24
Diagnostic Agent Driver	"Running Diagnostics" on page 26

Table 2: Linux Installation Highlights

Diagnostics

For detailed information about the diagnostic Agent, see chapter 4.

Terms

Table 3 lists the terms and acronyms used in this document.

Term	Description
Agent	Acceleration Agent (Anti-Virus, XML, Diagnostic, and the like)
Bitstream	A continuous flow of binary digits (bits), through some form of communication (in this case, the PCI bus), with no break or separations between the characters
CPC	Content Processing Controller: The controller on the CPP board that acts as a bridge or arbiter between the PCI bus, Agents, and DDR SDRAM
CPE	Content Processing Engine: One of two reconfigurable logic components on the CPP board. Each CPE supports multiple acceleration Agents.
CPP	Tarari's Content Processing Platform, and the board on which it is installed.
DDR	Double Data Rate memory: The type of memory on the CPP board
DMA	Direct Memory Access: A fast method of moving data directly to memory to speed processing
I/O	Input/Output
MSB	Most Significant Bit: That portion of a number, address, or field which occurs leftmost when its value is written as a single number in conventional binary or hexadecimal notation.
QWORD	A qword (quad word) pointer points to 8 bytes of data at a time. For example: Pointer X initially points to bytes 0-7. If it increments by 1, Pointer X then points to bytes 8-15.
SRAM	Static Random Access Memory: The low latency memory on the CPP board. Each CPE has two associated SRAM banks.
VHDL	Very High Speed Integrated Circuit (VHSIC) Hardware Description Language

Table 3: Terms Used in this Document

Notes

2 Windows Installation

Test Environment

For Linux installations, see chapter 3.

Table 4 describes the preferred test environment, used to validate the CPP board, its Agents, and the Windows drivers:

Requirement	Description
Hardware	The CPP board
	Windows 2000 x86
Platform	Windows 2000 Server, Service Pack 2
	1 GB RAM
	Dual Intel* Pentium 3 Processors, Intel STL2 Motherboard
	The CPP is the only board installed on the PCI bus

Table 4: Test Environment

Required Files

Table 5 lists the required files.

File Name	Description
config_tool.exe	The Agent Acceleration Set (AAS) Configuration Tool
Bit2Agent.exe	The Bitstream to Agent Utility
diagnostic.agt	The Diagnostic Agent
diag.bit	The Diagnostic Agent Xilinx* bitstream file

Table 5: Required Files

Installing the CPP Base Driver

Procedure

Once Windows starts, it automatically discovers new hardware, and runs the New Hardware Wizard.

1. Specify the directory that contains the `cpp.inf` and `cppW2K.sys` files. For example (using CD-ROM drive D:\):

   D:\redist\drivers

2. If Windows did not detect the new hardware, or if you skipped the New Hardware Wizard, start the Windows Control Panel and double-click **System**.
3. From the Hardware tab, click *Device Manager*.
4. In the *Other Devices* list, locate the unconfigured PCI board.
5. Right-click the unconfigured PCI board, and select **Properties**.
6. Click *Install Driver*.
7. Follow the wizard's instructions and specify the directory with the `cpp.inf` and `cppW2K.sys` files, as in step 1 above.
8. Restart your system.

Windows 2000 Configuration Setting

1. From the Windows Control Panel, double-click **Network and Dialup connections**.
2. Right-click the **Local Area Connection** and select **Properties**.
3. From the list, select **File and Printer Sharing for Microsoft Networks**.
4. Click *Properties*.
5. Under *Optimizations*, select **Maximize data throughput for network applications**.
6. Restart your system.

Using the Bitstream to Agent (Signing) Utility

NOTE: This procedure is required for both the Windows and Linux platforms.

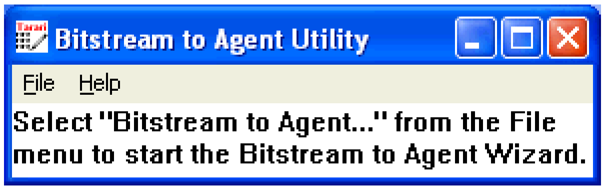
Procedure

Before you program the CPEs with Agents, you **must** first "sign" each bitstream file, by converting it into a format valid for the CPEs.

This utility appends Agent metadata to a Xilinx* bitstream file (`.bit` extension), then converts it to the Agent file (`.agt` extension) format the CPEs require. The utility can also display existing Agent file information.

1. From Windows Explorer, double-click the `Bit2Agent.exe` file.

   The utility's main window displays, as Figure 1 shows.



Select "Bitstream to Agent..." from the File menu to start the Bitstream to Agent Wizard.

Figure 1: Opening Message

2. From the *File* menu, select **Bitstream to Agent**, as Figure 2 shows.

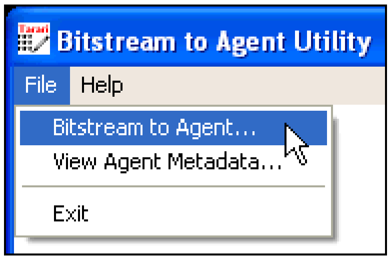
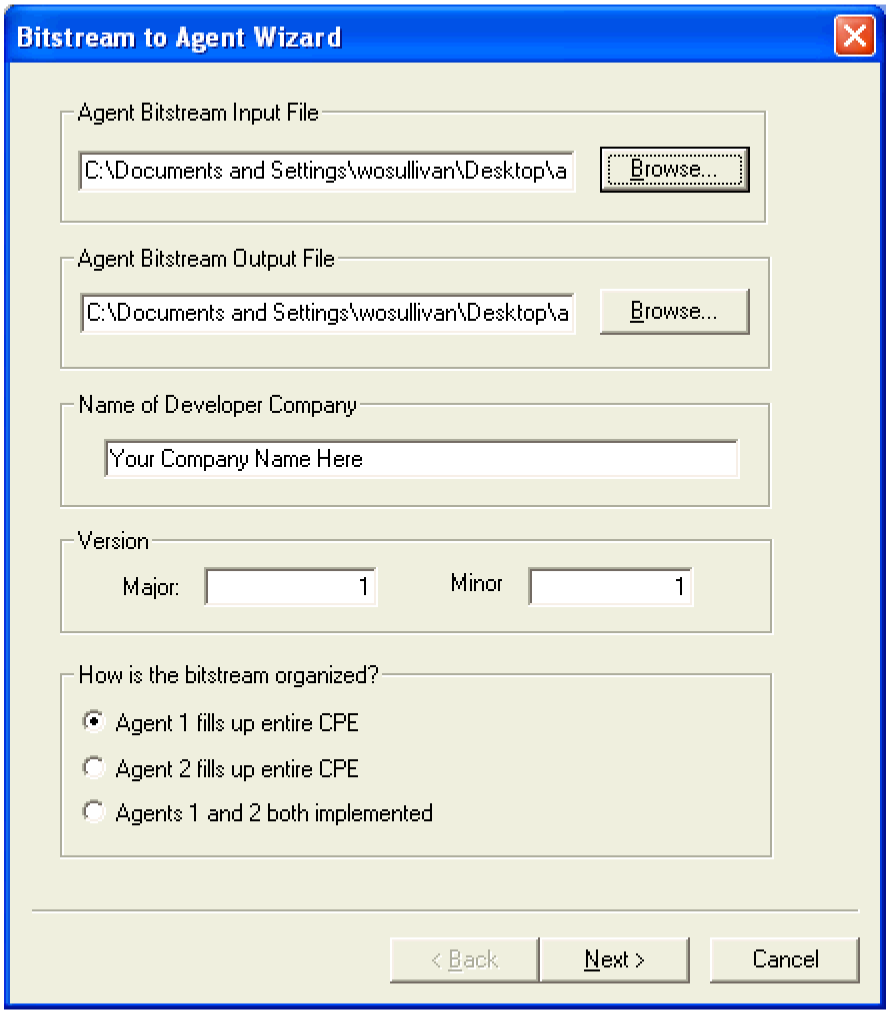


Figure 2: Starting the Conversion Process

The *Agent Specification Wizard* in Figure 3 displays.



Agent Bitstream Input File: C:\Documents and Settings\wosullivan\Desktop\a

Agent Bitstream Output File: C:\Documents and Settings\wosullivan\Desktop\a

Name of Developer Company: Your Company Name Here

Version

Major: 1 Minor 1

How is the bitstream organized?

- [✓ Agent 1 fills up entire CPE]
- Agent 2 fills up entire CPE
- Agents 1 and 2 both implemented

Figure 3: Agent Specification

3. Type or browse to the *Agent Bitstream Input File* (`.bit` extension).

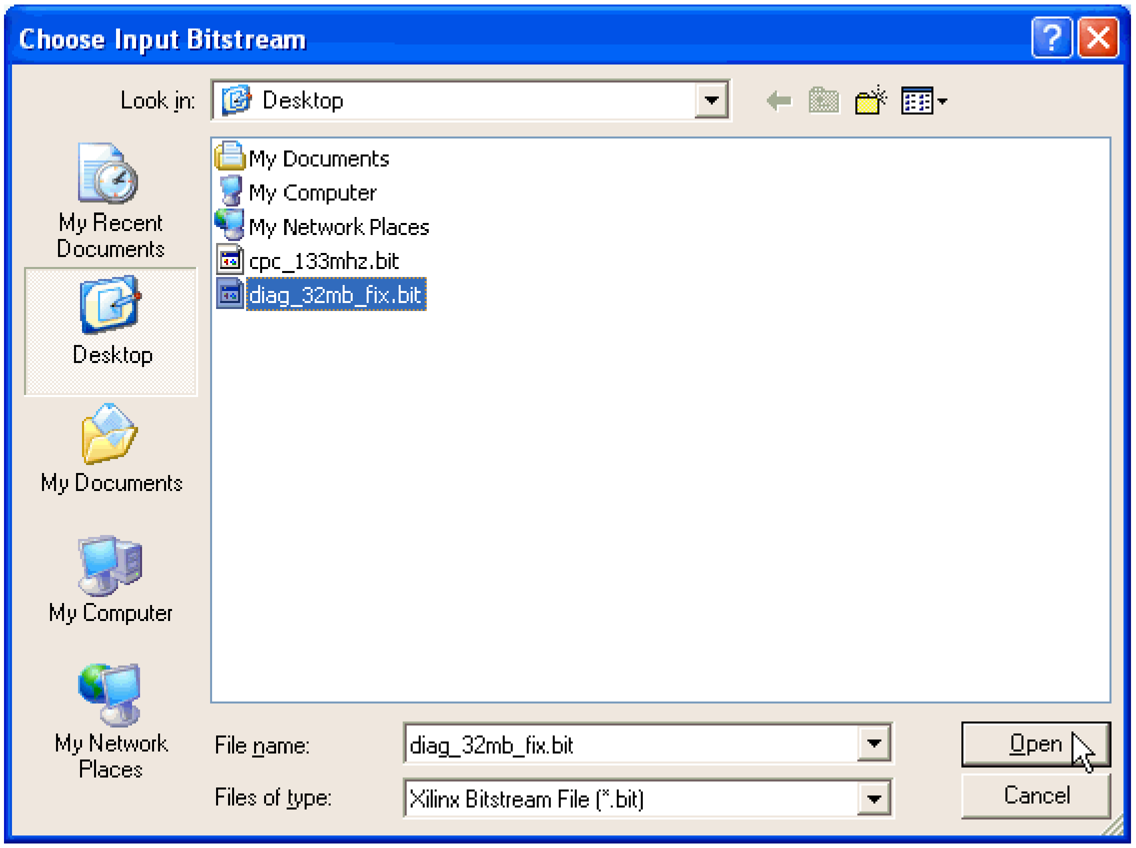
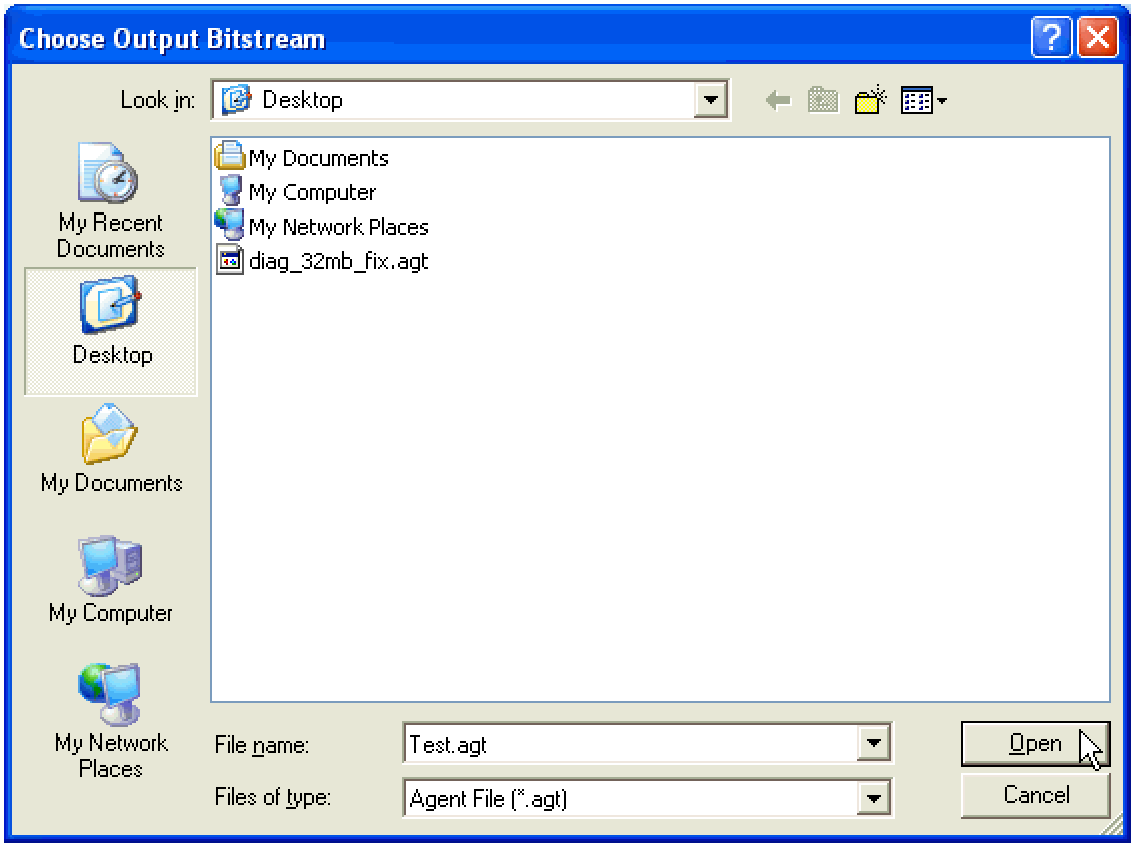


Figure 4: Input File Name

4. Type or browse to the *Agent Bitstream Output File*, or accept the default, which is the same path and file name, but with the `.agt` (Agent) extension.

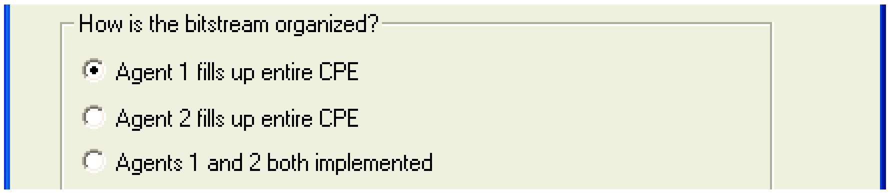


My Documents
My Computer
My Network Places
diag_32mb_fix.agt

File name: Test.agt
Files of type: Agent File (*.agt)

Figure 5: Output File Name

5. Type the *Name of Developer Company* and *Version* information if needed. Both the *Major* and *Minor* default version settings are **1**.
6. Make your selection, as Figure 6 shows. Information about each setting is listed in Table 6.



How is the bitstream organized?

(● Agent 1 fills up entire CPE)
○ Agent 2 fills up entire CPE
○ Agents 1 and 2 both implemented

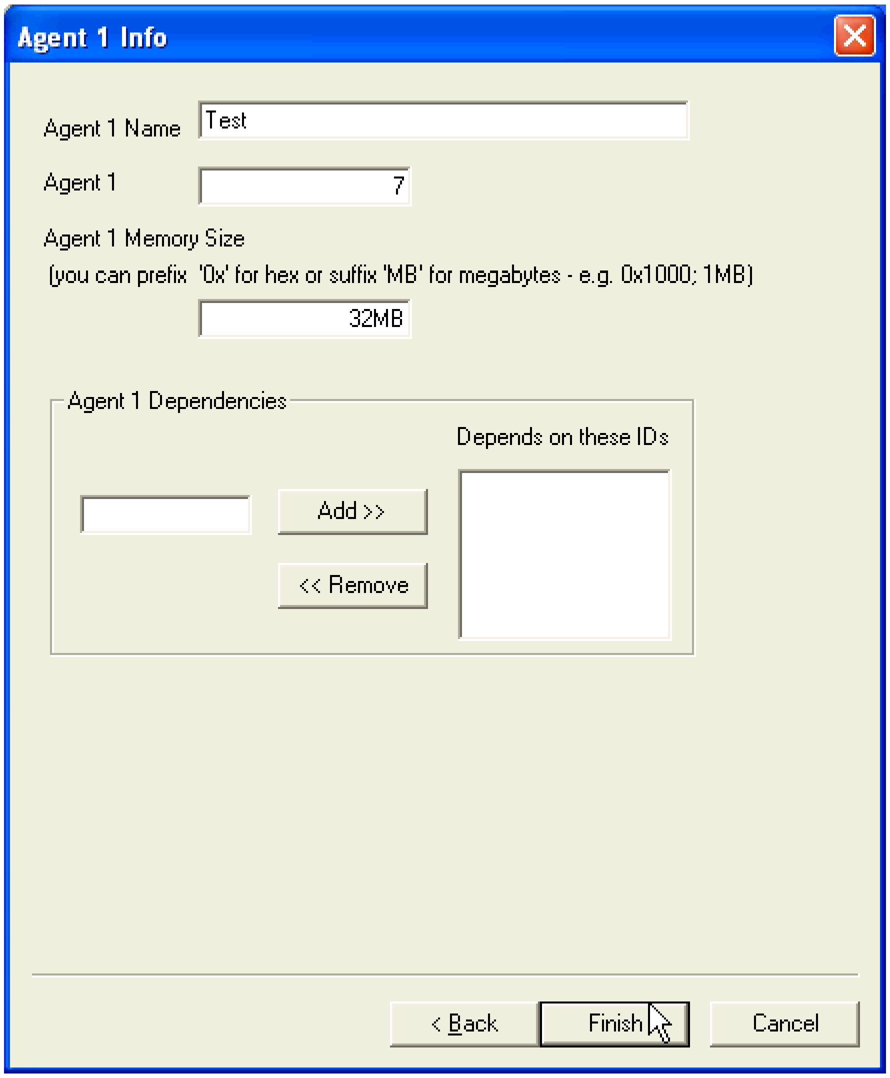
Figure 6: Bitstream Load Selection

Radio Button	Description
Agent 1 fills up entire CPE	Any Agent 1 file you load uses the full CPE memory space
Agent 2 fills up entire CPE	Any Agent 2 file you load uses the full CPE memory space
Agents 1 and 2 both implemented	Agents 1 and 2 load the same Agent file

Table 6: Agent Locations

7. Click *Next*.

The *Agent Info* dialog displays, as Figure 7 shows.



Agent 1 Name: Test
Agent 1: 7
Agent 1 Memory Size
(you can prefix '0x' for hex or suffix 'MB' for megabytes - e.g. 0x1000; 1MB): 32MB
Agent 1 Dependencies
Depends on these IDs

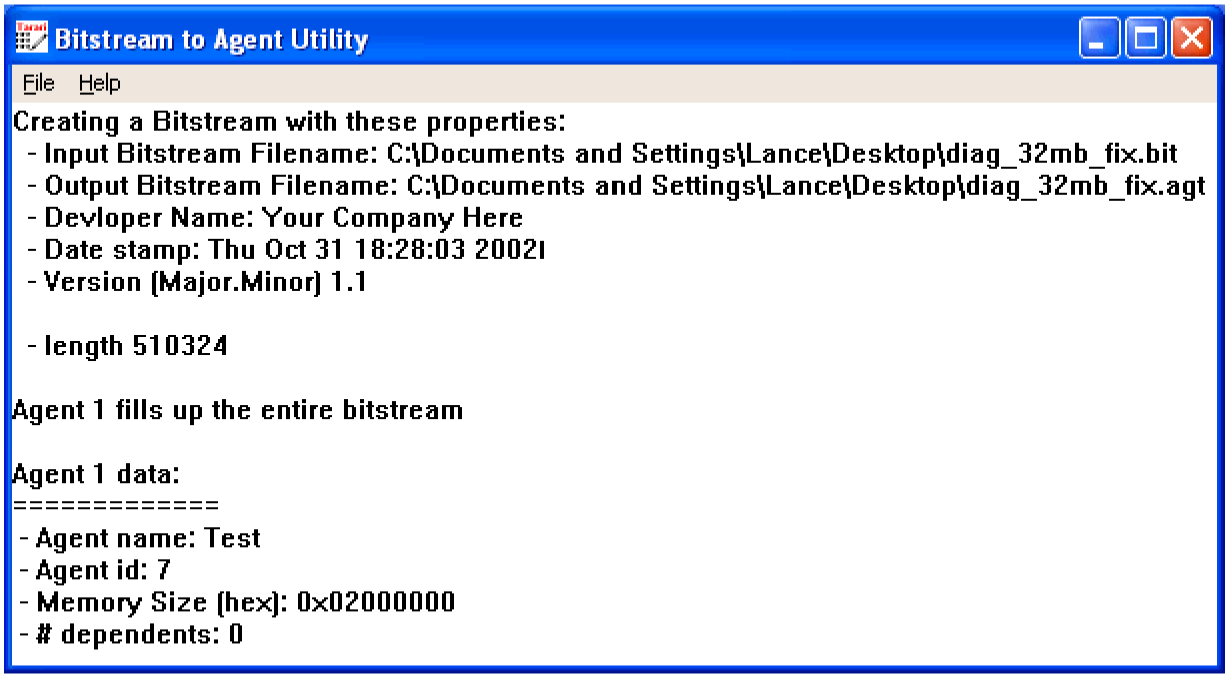
Figure 7: Agent Information Dialog

8. Type the *Agent 1 Name* in the text box. This is for information purposes only.
9. Type the *Agent 1 ID* in the text box. The Agent ID is unique to each Agent, and is the decimal equivalent of a 32-bit binary number, as specified in the corresponding Agent driver. This routes communication, such as interrupts and data, to and from the appropriate Agent and drivers. The Diagnostic Agent is hard coded to **1** in the driver. For information about the Agent ID numbering convention, see Appendix A.
10. Type the *Agent 1 Memory Size* in the text box. This specifies how much memory to reserve for the Agent. The Diagnostic Agent uses 1 MB.

    The Agent can depend on one or more additional Agents.
11. **Optional**: Type the ID of the Agent that this Agent depends on, then click *Add*. The Diagnostic Agent is not dependent on any other Agents.
12. Click *Finish*.
13. If defining two Agents, click *Next*.

14. If defining two Agents, type the information for Agent 2 and click *Finish*.

    Once you click *Finish*, the Agent summary displays, as Figure 8 shows.



```
Creating a Bitstream with these properties:
  - Input Bitstream Filename: C:\Documents and Settings\Lance\Desktop\diag_32mb_fix.bit
  - Output Bitstream Filename: C:\Documents and Settings\Lance\Desktop\diag_32mb_fix.agt
  - Devloper Name: Your Company Here
  - Date stamp: Thu Oct 31 18:28:03 2002I
  - Version (Major.Minor) 1.1

  - length 510324

Agent 1 fills up the entire bitstream

Agent 1 data:
=============
 - Agent name: Test
 - Agent id: 7
 - Memory Size (hex): 0x02000000
 - # dependents: 0
```

Figure 8: Agent Creation Properties

Viewing Existing Files

1. To view an existing Agent's metadata, from the *File menu, select View Agent MetaData*, as Figure 9 shows.

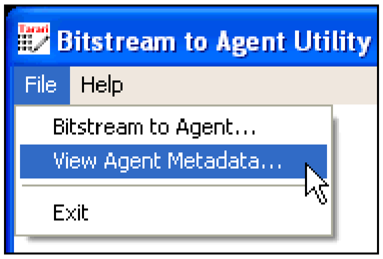
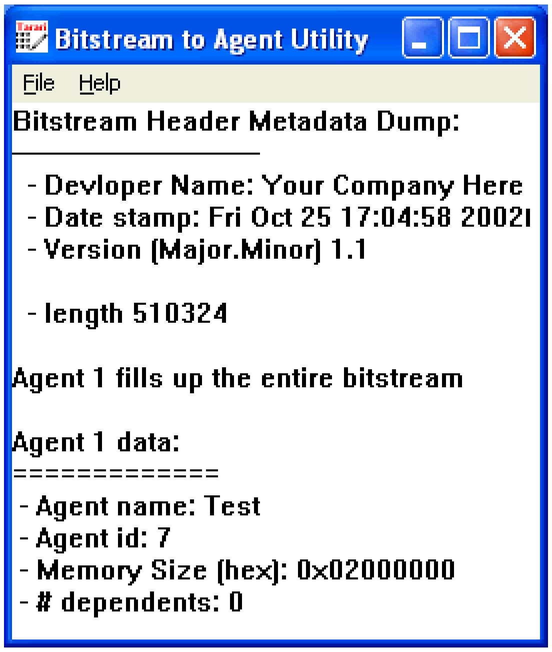


Figure 9: Menu Selection

The Agent summary information displays, as Figure 10 shows.



```
Bitstream Header Metadata Dump:

 - Devloper Name: Your Company Here
 - Date stamp: Fri Oct 25 17:04:58 2002I
 - Version (Major.Minor) 1.1

 - length 510324

Agent 1 fills up the entire bitstream

Agent 1 data:
=============
- Agent name: Test
- Agent id: 7
- Memory Size (hex): 0x02000000
- # dependents: 0
```

Figure 10: Agent File Information

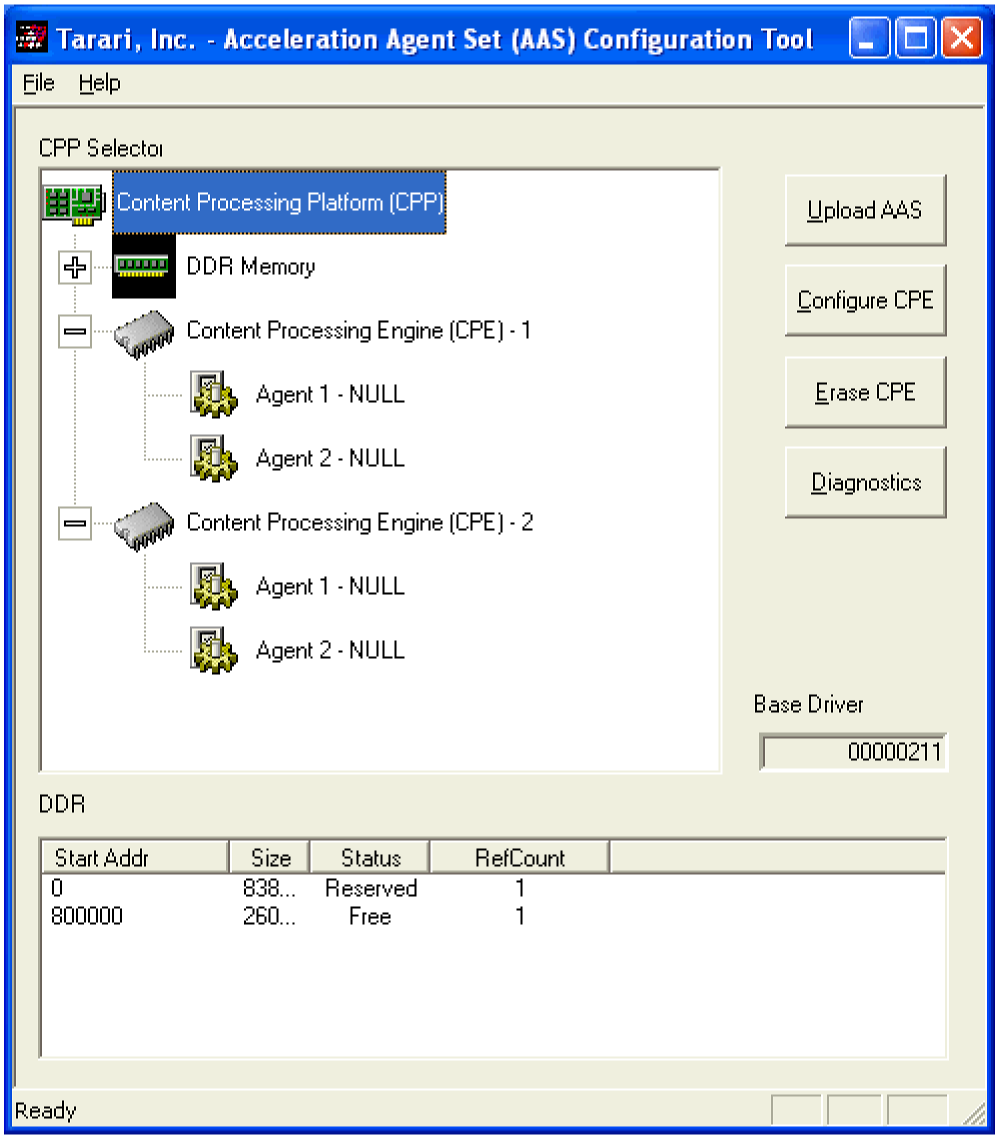
NOTE: If your system is on the Linux platform, return to chapter 3 for further instructions (see "Using the cpp_agent_manager Tool" on page 24 in chapter 3).

Using the Configuration Tool

To improve its usability, the Configuration Tool provides multiple methods to accomplish the same tasks. In addition to the methods this document describes, you can right-click on the *CPP*, *DDR Memory*, or either *CPE* to configure them. For ease of reading, this document describes only one method to accomplish each task.

1. From Windows Explorer, double-click the `config_tool.exe` application.

   The Agent Configuration Tool displays, as Figure 11 shows.


Tarari, Inc. - Acceleration Agent Set (AAS) Configuration Tool
File
Help
CPP Selector
Content Processing Platform (CPP)
DDR Memory
Content Processing Engine (CPE) - 1
Agent 1 - NULL
Agent 2 - NULL
Content Processing Engine (CPE) - 2
Agent 1 - NULL
Agent 2 - NULL
Upload AAS
Configure CPE
Erase CPE
Diagnostics
Base Driver
DDR
Ready

Base Driver: 00000211

Start Addr	Size	Status	RefCount
0	838...	Reserved	1
800000	260...	Free	1

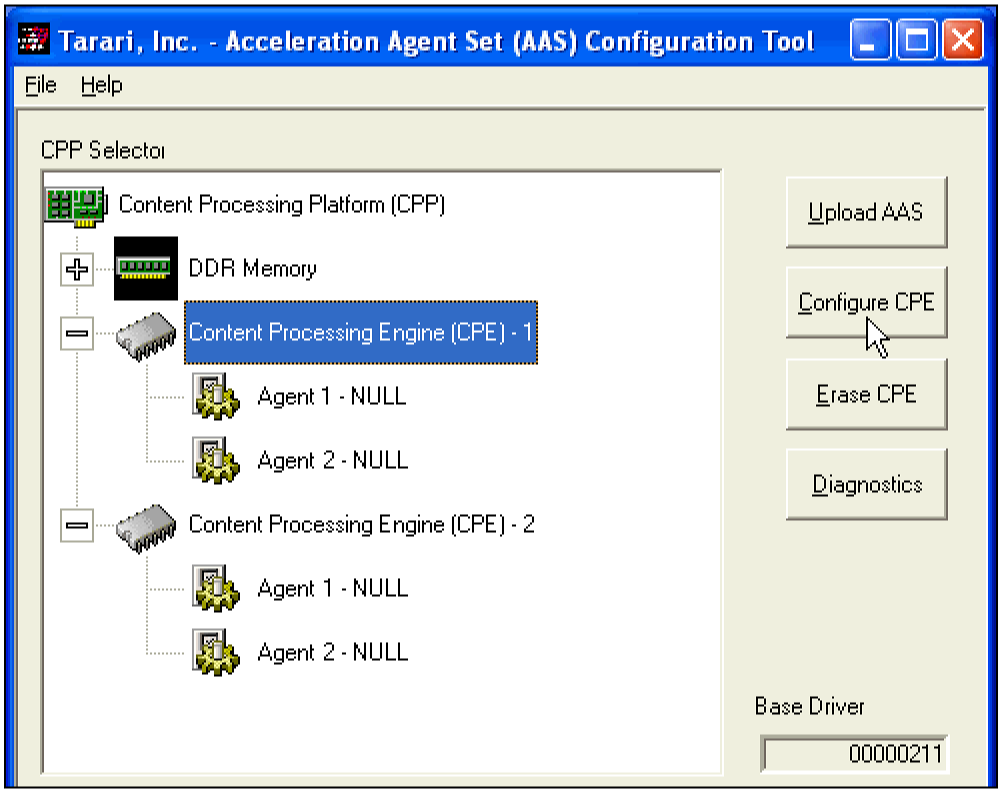
Figure 11: Agent Configuration Tool Dialog

Assigning the DDR Memory

Before you configure the CPEs to run Agents, you must first specify which of eight slots (numbered 0 through 7) in DDR memory will contain which Agents.

1. Click the plus sign (+) to the left of the *DDR Memory*.

   The *DDR Memory* display expands.

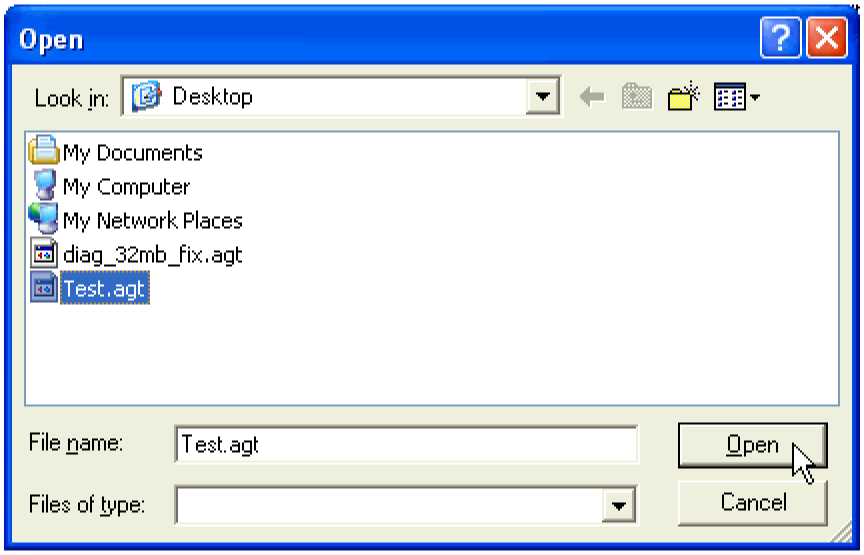


Content Processing Platform (CPP)
DDR Memory
Content Processing Engine (CPE) - 1
Agent 1 - NULL
Agent 2 - NULL
Content Processing Engine (CPE) - 2
Agent 1 - NULL
Agent 2 - NULL

Base Driver: 00000211

Figure 12: Assigning DDR Memory Slots

2. Click the first empty slot.
3. Click *Upload AAS* (Acceleration Agent Set).
4. Double-click the Agent file (`.agt` extension) to load into DDR memory.



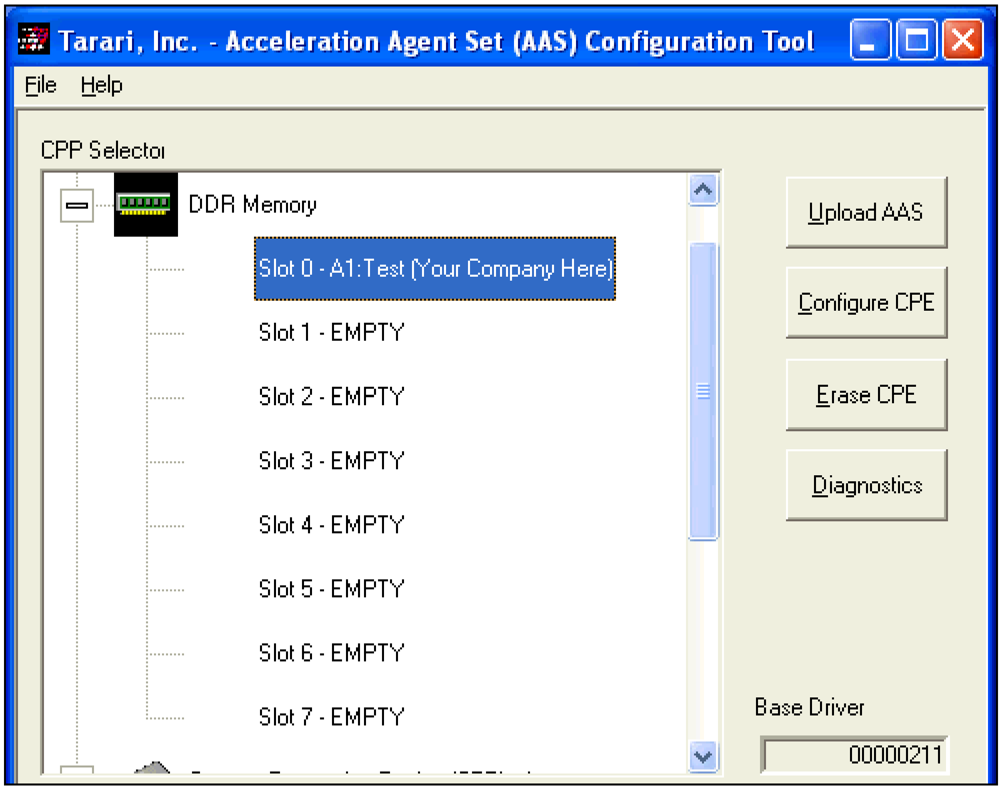
Look in: Desktop

My Documents
My Computer
My Network Places
diag_32mb_fix.agt
Test.agt

File name: Test.agt
Files of type:

Figure 13: Selecting an Agent to Load

The Agent loads into the DDR memory slot, as Figure 14 shows.



DDR Memory
Slot 0 - A1:Test (Your Company Here)
Slot 1 - EMPTY
Slot 2 - EMPTY
Slot 3 - EMPTY
Slot 4 - EMPTY
Slot 5 - EMPTY
Slot 6 - EMPTY
Slot 7 - EMPTY

Base Driver 00000211

Figure 14: Agent Loaded in DDR Memory Slot 1

5. Repeat steps 2 through 4 above for any additional Agents.

Configuring the CPEs

Follow this procedure to configure a CPE with an uploaded bitstream.

1. In the CPP Selector window, select a CPE (1 or 2) to configure.
2. Click *Configure CPE*.

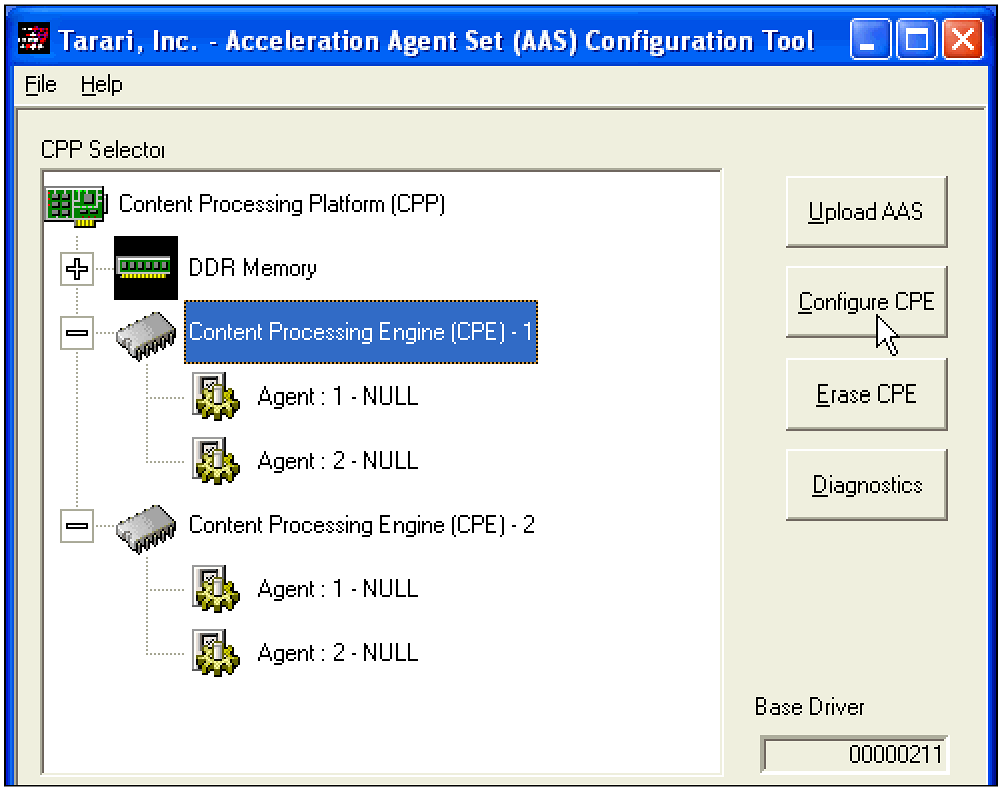

Tarari, Inc. - Acceleration Agent Set (AAS) Configuration Tool
File
Help
CPP Selector
Content Processing Platform (CPP)
DDR Memory
Content Processing Engine (CPE) - 1
Agent : 1 - NULL
Agent : 2 - NULL
Content Processing Engine (CPE) - 2
Agent : 1 - NULL
Agent : 2 - NULL
Upload AAS
Configure CPE
Erase CPE
Diagnostics
Base Driver
00000211

Figure 15: Configuring the CPEs

3. From the dialog (Figure 16), double-click the Agent to load onto the CPE.

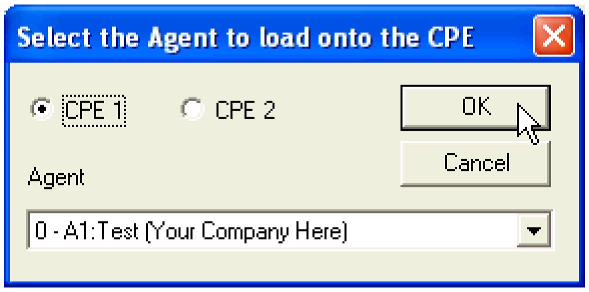
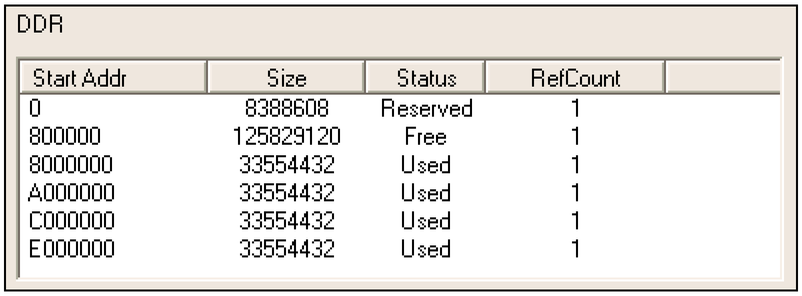

Select the Agent to load onto the CPE
CPE 1
CPE 2
OK
Cancel
Agent
0 - A1:Test (Your Company Here)

Figure 16: Loading an Agent onto the CPE

4. Repeat steps 1 through 3 above for any additional Agents.

Memory Table

The Agent Configuration Tool's memory table, at the bottom of the screen, displays the allocated DDR memory information, as Figure 17 shows. The DDR memory updates only when you configure a CPE with an Agent.


DDR

Start Addr	Size	Status	RefCount
0	8388608	Reserved	1
800000	125829120	Free	1
8000000	33554432	Used	1
A000000	33554432	Used	1
C000000	33554432	Used	1
E000000	33554432	Used	1

Figure 17: Memory Table

Erasing the CPEs

1. To erase the contents of either CPE, from the *CPP Selector* window, first click the CPE you want to erase.
2. Click *Erase CPE*. The Agent name returns to NULL.

Running Diagnostics

This procedure runs diagnostic tests on the CPP.

1. Click *Diagnostics*.
2. From the confirmation dialog, click *Yes*.

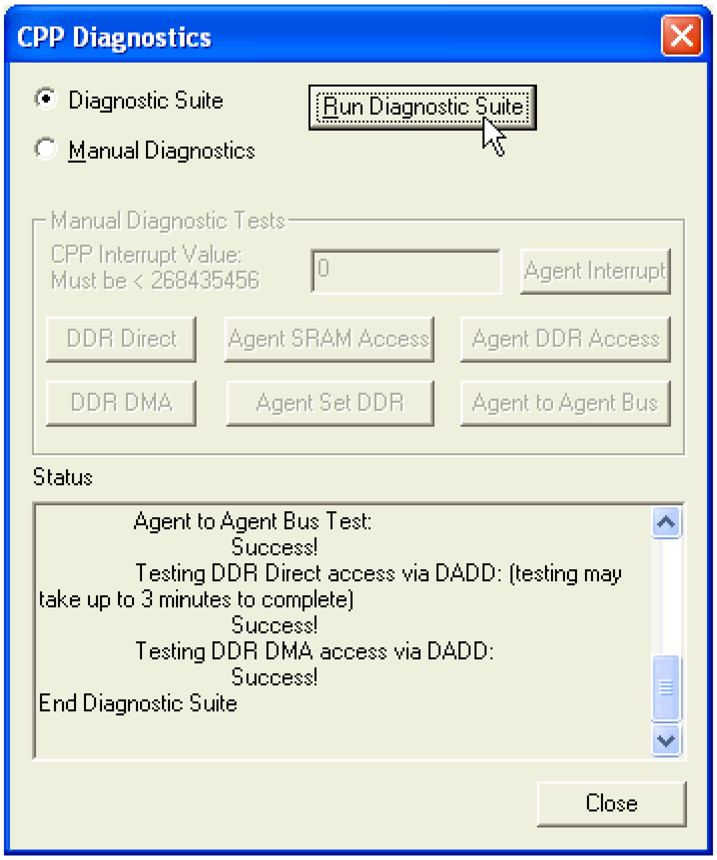
   This erases both CPEs, and loads them with the Diagnostic Agent.

   Once the Diagnostic Agent and driver are loaded and initialized, the Board Diagnostics dialog displays, as Figure 18 shows.

All Diagnostics

1. To run **all** diagnostics, select *Diagnostic Suite* (the default setting).
2. Click *Run Diagnostic Suite*.

   The test results display in the *Status* window.



```
              Agent to Agent Bus Test:
                              Success!
              Testing DDR Direct access via DADD: (testing may
take up to 3 minutes to complete)
                              Success!
              Testing DDR DMA access via DADD:
                              Success!
End Diagnostic Suite
```

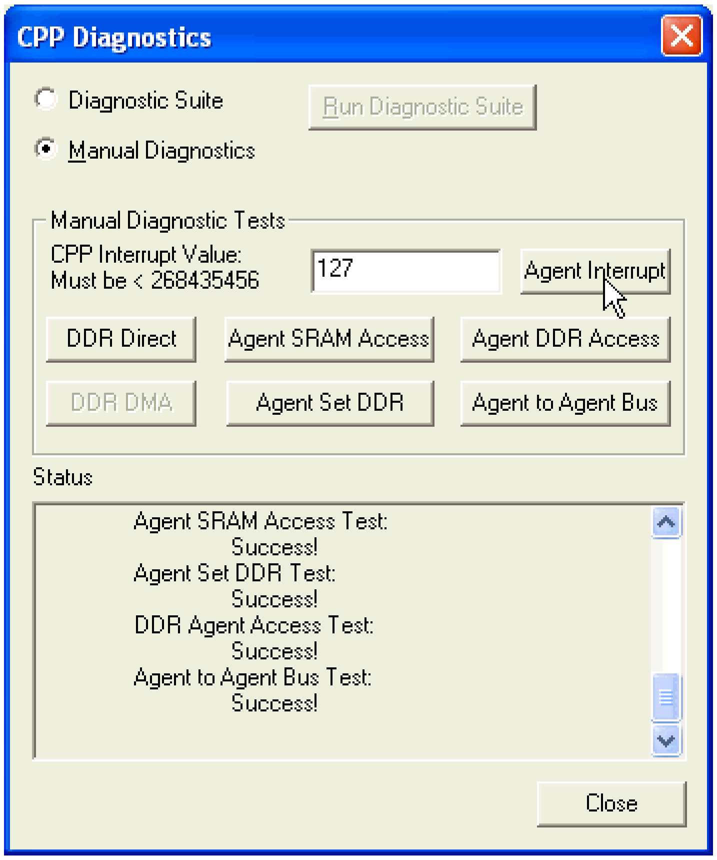
Figure 18: Running All Diagnostics

Specific Diagnostics

1. To run **specific** diagnostics, select *Manual Diagnostics*.

   This enables the Manual Diagnostic Tests.

NOTE: The DDR DMA test is currently disabled.



CPP Interrupt Value:
Must be < 268435456
127

```
Agent SRAM Access Test:
        Success!
Agent Set DDR Test:
        Success!
DDR Agent Access Test:
        Success!
Agent to Agent Bus Test:
        Success!
```

Figure 19: Running Manual Diagnostics

2. For the Agent Interrupt test, you must first type a number in the *Agent Interrupt Value* text box. The valid range is 0 (the default) to 268,435,456 (without the commas).
3. Click the test you want to run, as often as needed.

   The test results display in the *Status* window.
4. For specific information about the diagnostic tests themselves, see chapter 4.

3 Linux Installation

Test Environment

For Windows installations, see chapter 2.

Table 7 describes the preferred test environment, used to validate the CPP board

Requirement	Description
Hardware	The CPP board
	Intel* x86
Platform	Red Hat* Linux, version 7.2 or 7.3
	512 MB RAM
	Dual Intel* Pentium 3 Processors, Intel STL2 Motherboard
	The CPP is the only board installed on the PCI bus

Table 7: Test Environment

Required Files

Table 8 lists the required files.

File Name	Description
cpp_base-1.0.1-1.i386.rpm (or later)	The Base Driver and tools
cpp_diag-1.0.1-1.i386.rpm (or later)	The Diagnostic Agent and Driver
cpp_generic-1.0.1-1.i386.rpm	The Agent driver template

Table 8: Required Files

Installing the Base and Diagnostic Agent Drivers

This procedure describes the installation of the base driver and diagnostic Agent software. You **must** have administrative access privileges to load and manage these files.

1. Insert the Tarari floppy disk or CD-ROM.
2. If using the floppy disk, mount the floppy disk by typing:
   ```
   mount /mnt/floppy
   cd /mnt/floppy
   ```
3. Or, if using the CD-ROM, mount the CD-ROM by typing:
   ```
   mount /mnt/cdrom
   cd /mnt/cdrom
   ```
4. To list the file names, type:
   ```
   ls
   ```
   The base driver and diagnostic Agent file names display.
5. To install the base driver, type:
   ```
   rpm -i filename
   ```
   - *filename* is the full name of the base driver file (starts with `cpp_base`).
6. To install the diagnostic Agent, type:
   ```
   rpm -i filename
   ```
   - *filename* is the full name of the diagnostic Agent file (starts with `cpp_diag`).
7. To install the generic Agent driver template, type:
   ```
   rpm -i filename
   ```
   - *filename* is the full name of the generic Agent driver template (starts with cpp_generic).
8. To unmount the floppy disk, type:
   ```
   umount /mnt/floppy
   ```
9. Or, to unmount the CD-ROM, type:
   ```
   umount /mnt/cdrom
   ```
10. For additonal `rpm` command options, such as those used for displaying file names and locations, erasing files, and updating files, type:
    ```
    man rpm
    ```

Starting the Base and Diagnostic Agent Drivers

This procedure describes how to start the Base and Diagnostic Agent Drivers once you install them. This must be done before you proceed. If you like, you can include these commands in a script file. For more information, see "Running Diagnostics" on page 26.

1. To start the Base Driver, type:

   ```
   /usr/local/Tarari/cpp/base_driver/load_driver
   ```

2. To start the Diagnostic Agent Driver, type:

   ```
   /usr/local/Tarari/cpp/diag_driver/load_driver
   ```

Unloading the Base and Diagnostic Agent Drivers

This procedure describes how to unload the Base and Diagnostic Agent Drivers, if needed. The drivers must be unloaded in the reverse order that they were started.

1. To unload the Diagnostic Agent Driver, type:

   ```
   /usr/local/Tarari/cpp/diag_driver/unload_driver
   ```

2. To unload the Base Driver, type:

   ```
   /usr/local/Tarari/cpp/base_driver/unload_driver
   ```

Using the Bitstream to Agent (Signing) Utility

Before you program the CPEs with Agents, you **must** first "sign" each bitstream file, by converting it into a format valid for the CPEs.

You must use the Bitstream to Agent Utility to perform this conversion. The utility is Windows-based, with a graphical user interface.

1. Perform your conversions using the Bitstream to Agent Utility. For more information, see "Using the Bitstream to Agent (Signing) Utility" on page 7 in chapter 2.
2. Once you finish your conversions, proceed to the next page in this chapter for information about the `cpp_agent_manger` tool, and running diagnostics, using Linux.

Using the cpp_agent_manager Tool

The CPP board does not have a GUI for Linux. However, you can perform all CPP board functions by typing the `cpp_agent_manager` command on the Linux command line, as described in Table 9.

For scripting convenience, all operations result in an exit code of `0` for success, or `1` for failure.

To display the usage dump information, type the `cpp_agent_manager` command without any switches.

Switches	Parameters	Description
Board (-b #)	Description	If you installed multiple CPP boards, this specifies which board (numbered 0 to 3) to target.
	Required switches	none
	Notes	Not required if you have only one CPP board installed (defaults to 0)
Info (-i)	Description	Displays information about the bitstream currently in a device memory slot
	Required switches	-s #

	Notes	This command displays the bitstream information for memory slot 2 on CPP board 0:
		`cpp_agent_manager -i -s 2`
		Example response:
		`Bitstream info:`
		`---------------`
		`Date: Wed Nov  6 15:44:46 2002`
		`Description: A1:Diag A2:Test Agent (Tarari)`
		`version: 1.1`
		`length: 510324`
		A1 and A2 designate agents 1 and 2, respectively, and the company name is in parentheses.

Write (-w *filename*)	Description	Uploads the bitstream contents of *filename* to a designated memory slot, and optionally verifies (using the -v switch) that the write operation worked correctly, by reading back the data that was just written.
	Required switches	-s # (-v is optional)
	Notes	This command writes the `foo.agt` bitstream to memory slot 2 on CPP board 1, and verifies the operation: `cpp_agent_manager -w foo.agt -s 2 -b 1 -v` If the write operation fails, it gives the number of differences encountered.

Table 9: Switches for cpp_agent_manager

Switches	Parameters	Description
Read (-r *filename*)	Description	Reads a bitstream from the designated memory slot, then writes it to the designated *filename*. This is the full, signed bitstream file, including the header, and is identical to the original file uploaded to the memory slot.
	Required switches	-s #
	Notes	This command writes contents of memory slot 2 to the `foo_read_back.agt` file: `cpp_agent_manager -r foo_read_back.agt -s 2`
Program (-p)	Description	Programs the specified CPE (0 or 1) with the bitstream that was uploaded to the memory slot using the write (-w) switch
	Required switches	-f # and -s #
	Notes	This command configures CPE 1 with the bitstream loaded in memory slot 2: `cpp_agent_manager -p -s 2 -f 1`
Slot (-s #)	Description	Specifies the memory slot number (0 to 7)
	Required switches	none
	Notes	This switch is required for all other switches except the board (-b #).
CPE (-f #)	Description	Specifies the CPE (0 or 1)
	Required switches	none
	Notes	This switch is required for programming. See the program (-p) switch for more information.
Verify (-v)	Description	Verifies a write operation
	Required switches	none
	Notes	See the write (-w) switch for more information.
Help (-?)	Description	Displays this menu
	Required switches	none
	Notes	none

Table 9: Switches for cpp_agent_manager (continued)

Running Diagnostics

This procedure loads and runs the diagnostics. For this to work, you must first start the Base and Diagnostic Agent drivers. For more information, see "Starting the Base and Diagnostic Agent Drivers" on page 23.

1. If you want, you can insert any of these commands into a startup script, such as:

   ```
   /etc/rc.d/rc.local
   ```

2. To load the diagnostic Agent into memory, type:

   ```
   cpp_agent_manager -w /usr/local/Tarari/cpp/diag_driver/
     diagnostic.agt -s 0
   ```

   This command writes the `diagnostic.agt` file into DDR memory slot 0. The memory slot range is 0 to 7.

3. To program a CPE with the diagnostic Agent, type:

   ```
   cpp_agent_manager -p -s 0 -f 0
   ```

   This command programs CPE 0 with the file in DDR memory slot 0 (`diagnostic.agt`).

4. To also program the second CPE with the diagnostic Agent, type:

   ```
   cpp_agent_manager -p -s 0 -f 1
   ```

   This command programs CPE 1 with the file in DDR memory slot 0 (`diagnostic.agt`).

5. To run the diagnostics, type:

   ```
   cpp_diagnostics
   ```

   The diagnostics results display on-screen. If you type this command without having first loaded at least one CPE with the diagnostic agent, the CPP displays this error message:

   ```
   no agents available
   ```

4 Diagnostics and LEDs

Using the Diagnostic Agent

When you run Diagnostics (Windows or Linux), the Diagnostic Agent performs these functions:

- Receives Data on the I/O port from the Content Processing Controller (CPC)
- Sends Data on the I/O port Agent to the CPC
- Generates Interrupts
- Tests DDR memory
- Tests SRAM memory

I/O Register

Bits 31-28	0x8
Bits 27-0	Quad-word (QWord) aligned base address

Initially, the host sends an I/O command to the CPC with its Most Significant Bit (MSB) set to 1, which initializes the Agent. The Agent ignores all other I/O commands until initialization. Immediately after initialization, the Agent does an I/O write to the CPC with the value 0xBABEFACE.

The current application sets the address space for all Agents to 0. Each Agent uses the same 32 MB address space. Future versions will likely use separate 32 MB blocks of DDR memory.

Any I/O command sent before an ongoing test completes is ignored, except Initialize (MSB = 1), which is treated as a reset. The current test aborts, the Agent is initialized, and it waits for the next I/O command.

Table 10 describes the diagnostic tests.

Diagnostic Agent	Parameter	Description
Agent DDR Access	Purpose	Writes a pseudo-random QWord to each address in the Agent's DDR memory, then reads it back for comparison. It generates: • Pass (0x3), or • Fail (0x4), and the failed DDR memory address (byte) The current version seeds with 0x0 by default, but can be user-specified in future software versions.
	LEDs (CPE 1)	• D12 lights for Agent 1 if successful • D7 lights for Agent 2 if successful
	LEDs (CPE 2)	• D32 lights for Agent 1 if successful • D22 lights for Agent 2 if successful
	I/O Register Bits	• 31-28: 0x3 • 27-0: Random data seed
Agent Interrupt	Purpose	Specifies a value to be returned with the success interrupt (0x1), plus any interrupt data. For example: • 0x2000FFFF responds with 0x1000FFFF • 0x21234567 responds with 0x11234567
	LEDs	No LEDs light during this test.
	I/O Register Bits	• 31-28: 0x2 • 27-0: Interrupt data
Agent-to-Agent Bus	Purpose	Tests the Agent-to-Agent (A2A) bus, and generates these interrupts: • Success: 0x9 • Failure: 0xA • Testing: 0xB
	LEDs (CPE 1)	• D10 and D9 light for Agent 1 if successful
	LEDs (CPE 2)	• D28 and D26 light for Agent 1 if successful
	I/O Register Bits	• 31-28: 0x5 • 27-0: not defined
Clear DDR	Purpose	Fills the Agent's DDR memory with 0xF plus the fill data from the I/O command. It generates an interrupt value of 0x8 when the operation completes. The current version uses 0 for the fill data by default, but can be user-specified in future versions.
	LEDs	No LEDs light during this test.
	I/O Register Bits	• 31-28: 0x1 • 27-0: Fill data

Table 10: Diagnostic Agents

Diagnostic Agent	Parameter	Description
DDR Direct	Purpose	Tests the DDR memory direct access mechanism over the 256 MB range of DDR for this device. A predetermined pattern is written to the DDR, then read back. This test can run for up to three minutes, and the screen does not update until the test completes.
	LEDs	No LEDs light during this test.
	I/O Register Bits	• 31-28: not defined • 27-0: not defined
DDR DMA (currently disabled)	Purpose	Tests the DMA controller over the 256 MB range of DDR memory for this device.
	LEDs	No LEDs light during this test.
	I/O Register Bits	31-28: not defined 27-0: not defined
SRAM Access	Purpose	Writes a pseudo-random 18-bit word to each address in the Agent's SRAM, then reads it back for comparison. It generates either: • Success (0x5) • Failure (0x6), and the failed SRAM address (word) The current version seeds with 0x0 by default, but can be user-specified in future software versions.
	LEDs (CPE 1)	• D11 lights for Agent 1 if successful • D6 lights for Agent 2 if successful
	LEDs (CPE 2)	• D30 lights for Agent 1 if successful • D21 lights for Agent 2 if successful
	I/O Register Bits	• 31-28: 0x4 • 27-0: Random data seed

Table 10: Diagnostic Agents (continued)

Using the LEDs

Figure 20 shows the power LEDs on the front of the CPP board.

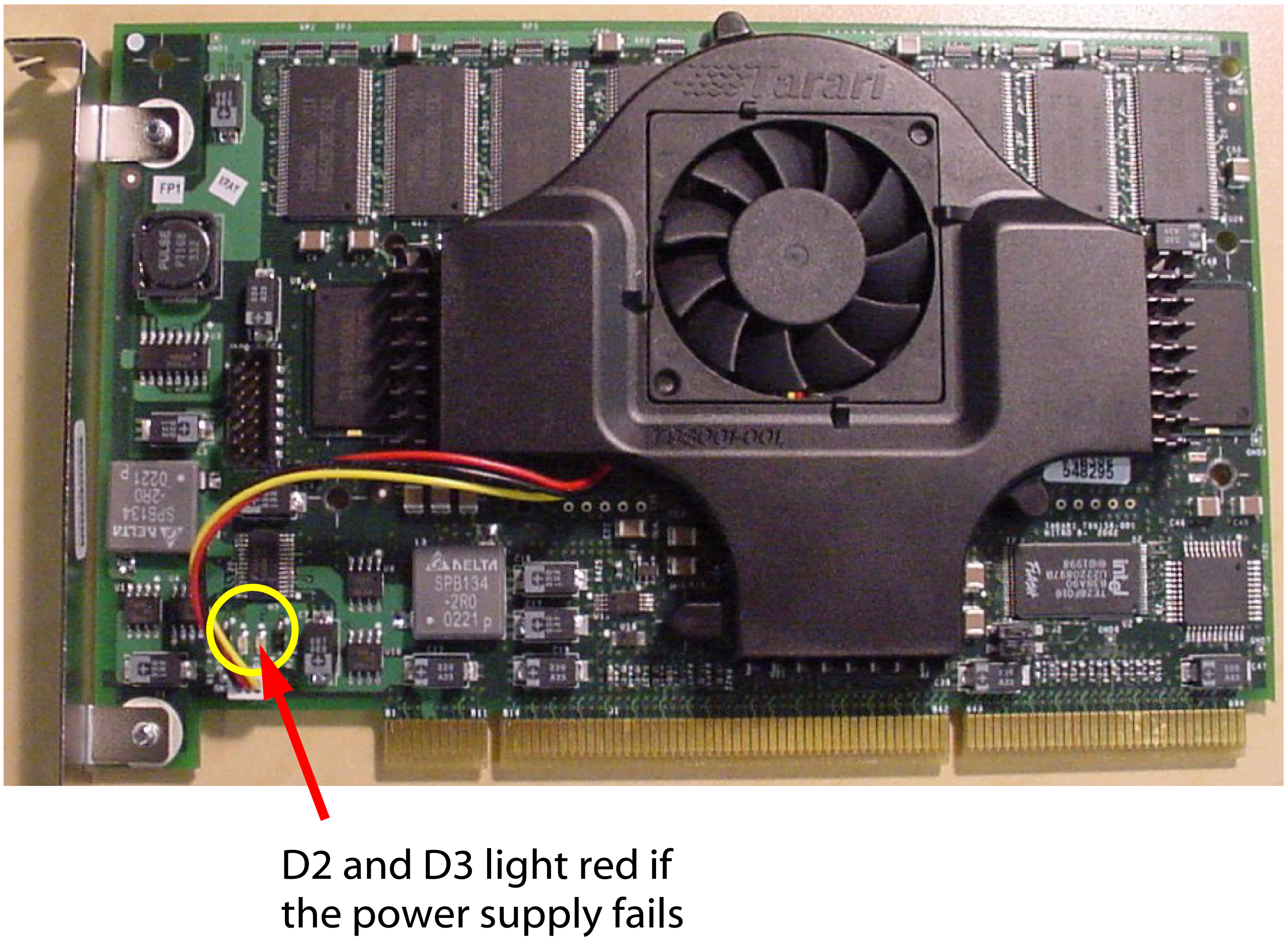


Figure 20: Power Supply LED Locations

Figure 21 shows the LEDs on the rear of the the CPP board.

To clear the LEDs after a test, you must erase the Diagnostic Agent from the CPE.

CPE 2

D34	🟨	Agent 1 initialized
D32	🟨	Agent 1 DDR test success
D30	🟨	Agent 1 SRAM test success
D28	🟨	Agent 1 A2A test success
D26	🟨	Agent 1 A2A slave
D24	🟨	Agent 2 initialized
D22	🟨	Agent 2 DDR test success
D21	🟨	Agent 2 SRAM test success
D20	🟩	Agent loaded successfully

CPE 1

D13	🟨	Agent 1 initialized
D12	🟨	Agent 1 DDR test success
D11	🟨	Agent 1 SRAM test success
D10	🟨	Agent 1 A2A test success
D9	🟨	Agent 1 A2A slave
D8	🟨	Agent 2 initialized
D7	🟨	Agent 2 DDR test success
D6	🟨	Agent 2 SRAM test success
D5	🟩	Agent loaded successfully

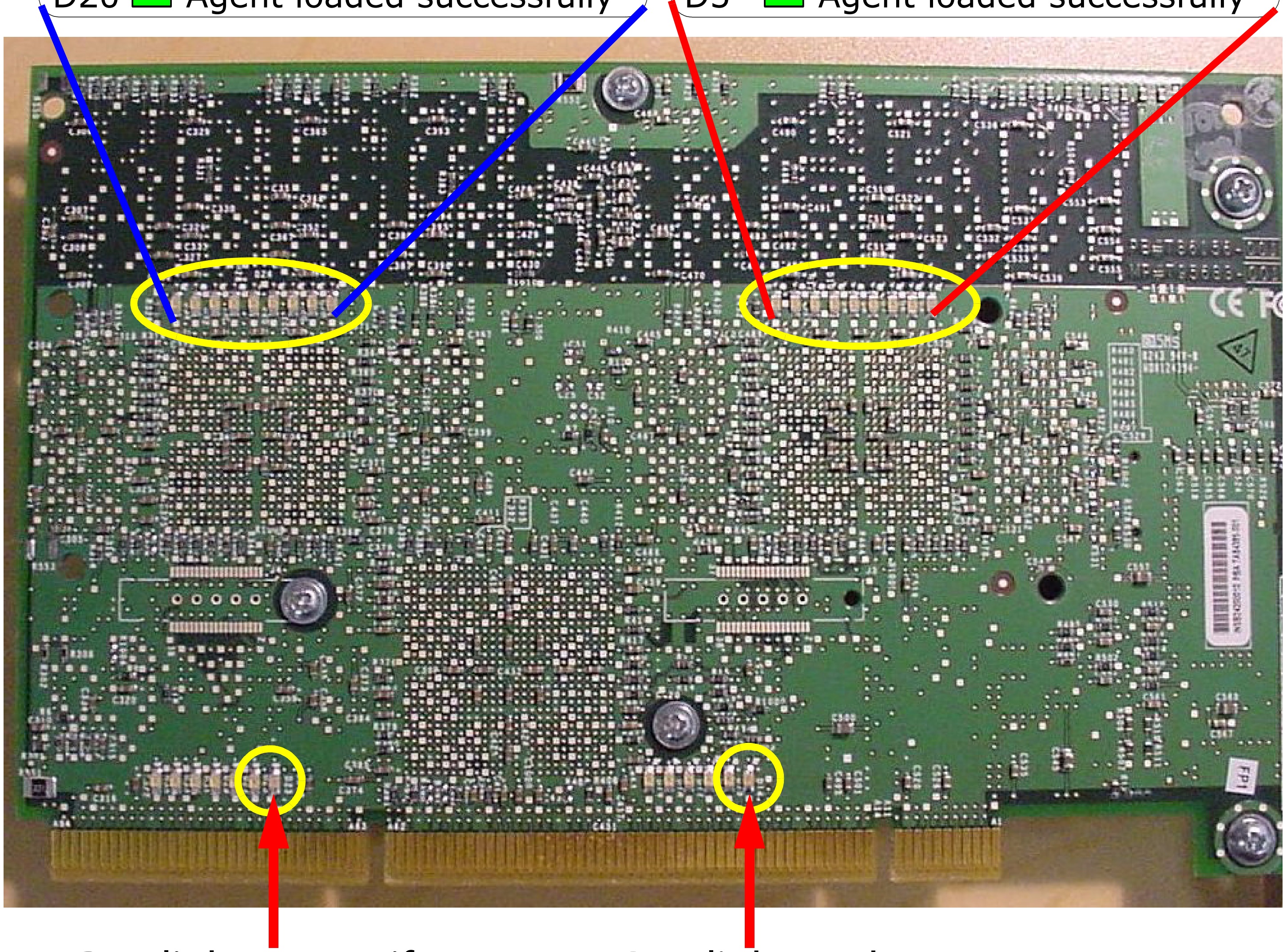

D23 lights green if the CPC is configured
D14 lights red during a reset

Figure 21: Diagnostic LED Location and Test Indication Summary

Notes

5 Flash Updates

Overview

This chapter describes how to update the 2 MB Flash memory on the CPP board, using Linux. You only need to follow this procedure if Tarari releases driver updates.

Updating the Flash Memory

Procedure

1. Retrieve the upgrade file, typically named `top.bit` or `cpc.bit`. The file name must end with the `.bit` extension.
2. Shut down your computer.
3. Open the computer case.
4. On the CPP board, install a 0.1 inch jumper across pins 1 and 2 of J2, as Figure 22 shows.

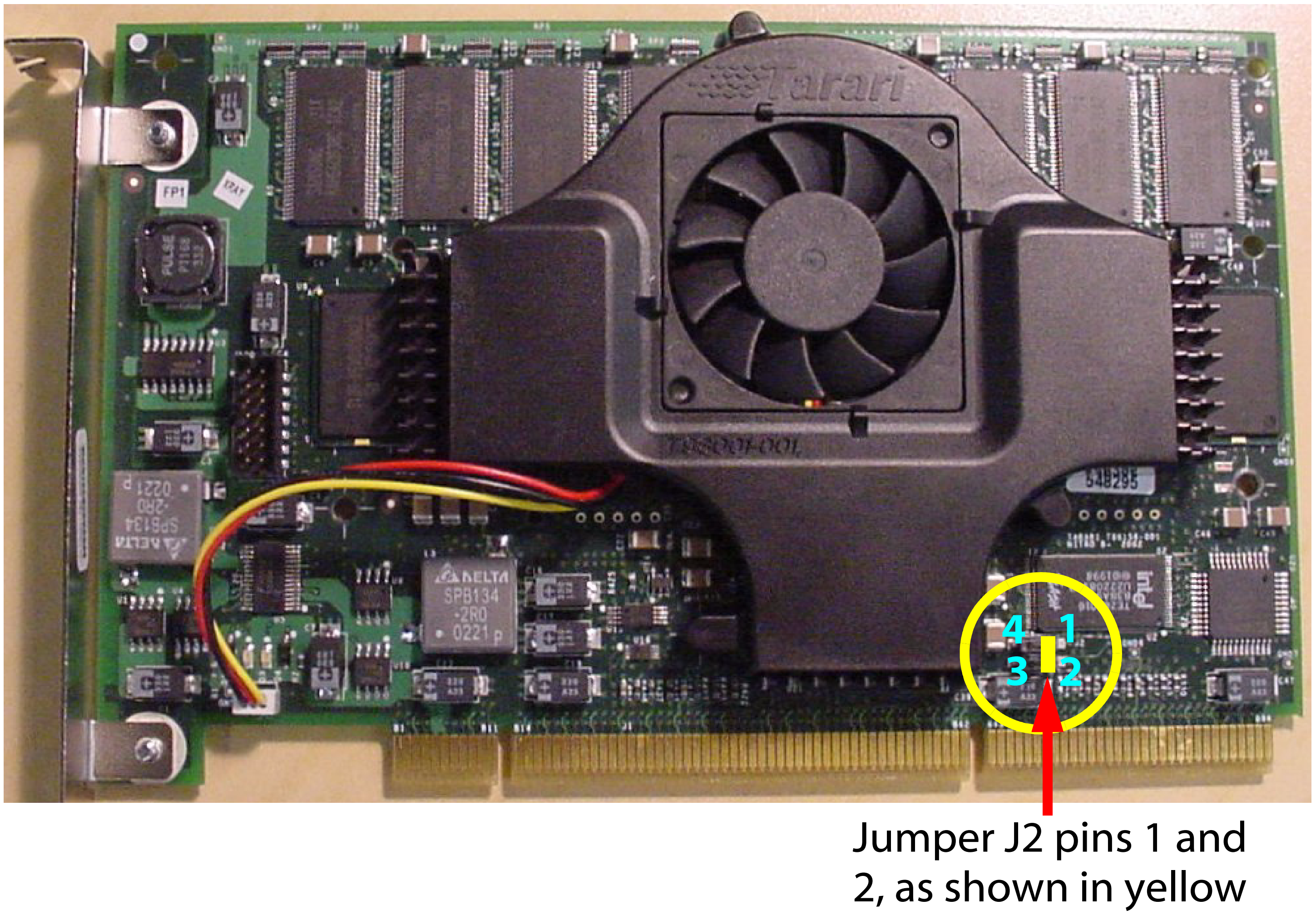


Figure 22: Connector J2 Location

5. Turn on your computer, and boot to Linux.
6. Type this command on the Linux command line:

```
cd /usr/local/Tarari/cpp/bin
```

7. Execute the Flash utility by typing this command:

```
cpp_flash_update
```

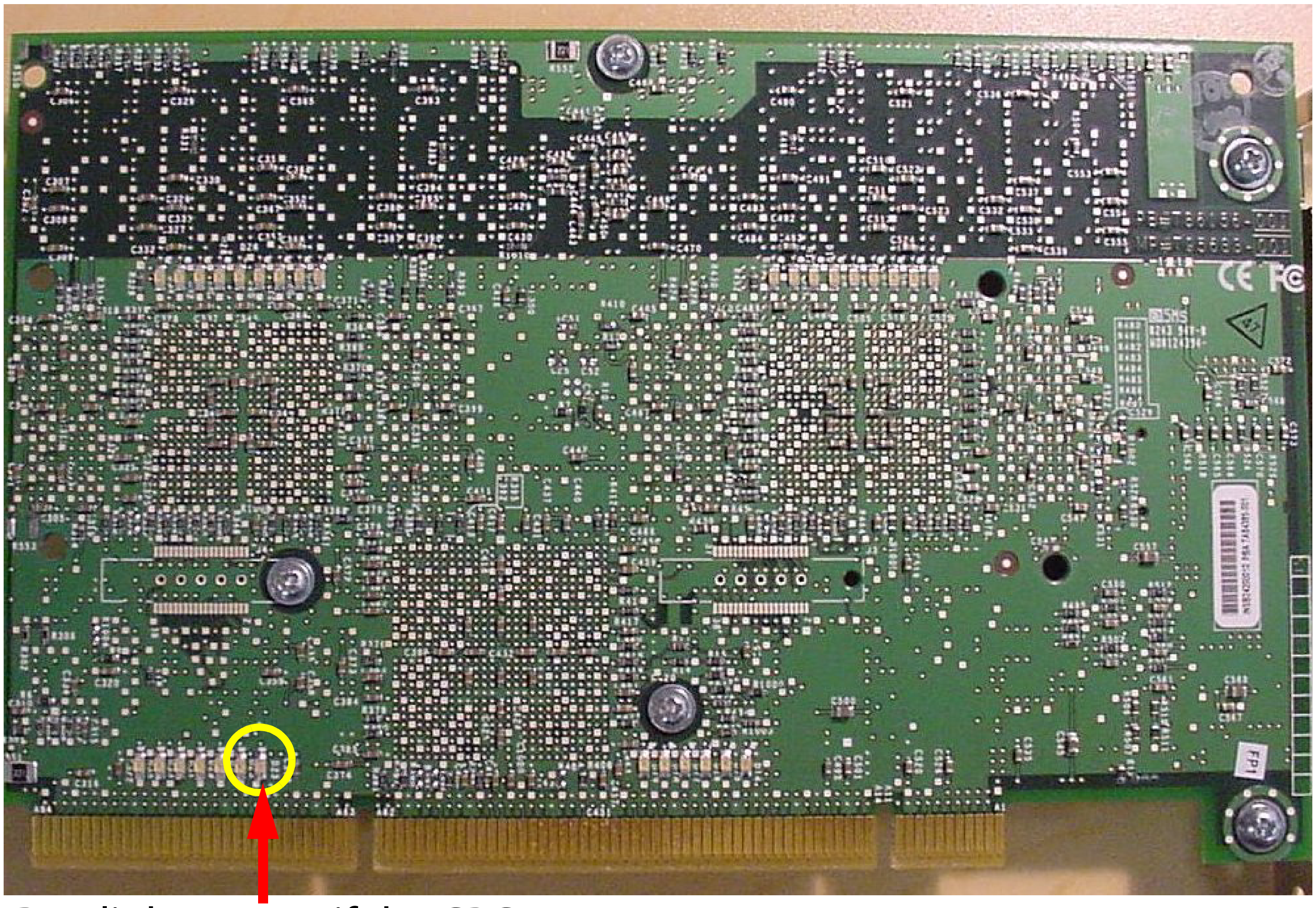
You installed this utility when you installed the base driver in chapter 3.

8. When the Flash utility prompts you for the file name, type the file name from step 1 above. The update process begins.
9. Wait for the process to complete. This takes about a minute. Do **not** turn off your computer during this time.
10. If the update fails, or if you accidentally interrupt the process, proceed to "If the Flash Update Fails" below.
11. Shut down your computer.
12. Remove the jumper that you installed on pins 1 and 2 of connector J2.
13. Secure the computer case.
14. Turn on the computer, and use the upgraded CPP board as you normally do.

If the Flash Update Fails

If the programming process encounters a problem, or is interrupted, the Flash memory is corrupted, causing the CPP board to not power up correctly.

After a power cycle, LED D23 on the rear of the CPP board does not light green if the update failed, as Figure 23 shows.



D23 lights green if the CPC is properly configured

Figure 23: LED D23 Location

To fix this problem, you must rerun the Flash utility to properly program the CPC with a backup image that resides in half of the Flash memory. To load the CPC from the backup image, follow this procedure:

1. Shut down your computer.
1. Jumper pins 3 and pin 4 of connector J2. Do **not** remove the jumper from pins 1 and 2.
2. Power on your computer and boot to Linux.
3. Run the flash utility, with **both** jumpers installed on J2.
4. When the process completes, shut down your computer.
5. Remove both jumpers from connector J2.
6. Secure the computer case.
7. Turn on the computer, and use the upgraded CPP board as you normally do.

A Reference

Agent ID Numbering Convention

When you use the Bitstream to Agent (Signing) Utility, you must assign each Agent a unique Agent ID number. For more information, see “Using the Bitstream to Agent (Signing) Utility” on page 7.

This Agent ID number used by the Bitstream to Agent Utility is the decimal equivalent of a 32-bit binary number. We define the bits from the most significant bit (MSB), and descending, or from left to right. Figure 24 shows an example of a Tarari RSA Security Agent.

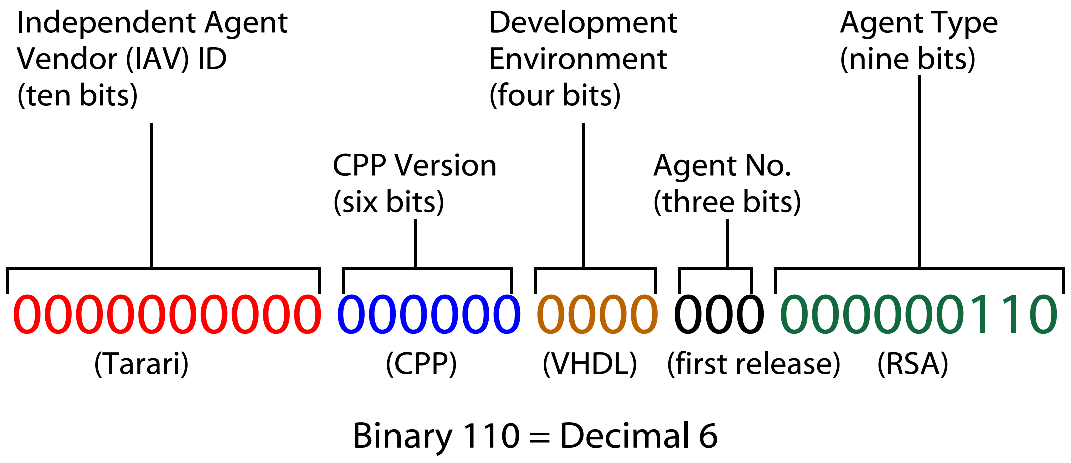


Figure 24: Agent ID Numbering

Table 11 on the next page lists the values for each set of bits.

Bit Definition	Parameter	Description
Independent Agent (IAV) ID	Number of bits	Ten (31:22)
	Assigned by	Tarari, or potentially a Tarari IAV development partner, such as Celoxica*
	Values	• 0000000000 Tarari • 0000000001 Celoxica • 0000000010 Reserved (to 1111111111)
CPP Version	Number of bits	Six (21:16)
	Assigned by	Tarari (the hardware Agent runs on this platform)
	Values	• 000000 CPP (Content Processing Platform) • 000001 Reserved (to 111111)
Development Environment	Number of bits	Four (15:12)
	Assigned by	Tarari
	Values	• 0000 VHDL • 0001 Verilog* • 0010 Celoxica • 0011 Reserved (to 1111)
Agent Number	Number of bits	Three (11:9)
	Assigned by	Vendor
	Values	• 000 First Release • 001 Second Release • 010 and so on...
Agent Type	Number of bits	Nine (8:0)
	Suggested by	Tarari
	Values	• 000000000 NULL • 000000001 Diagnostic Agent • 000000010 Reserved (to 111111111)

Table 11: Agent ID Numbering List

Index